Лабораторна робота № 3. Графові СУБД: Neo4j

МЕТА РОБОТИ: отримати навички роботи з графовими базами даних.

ПОСТАНОВКА ЗАДАЧІ

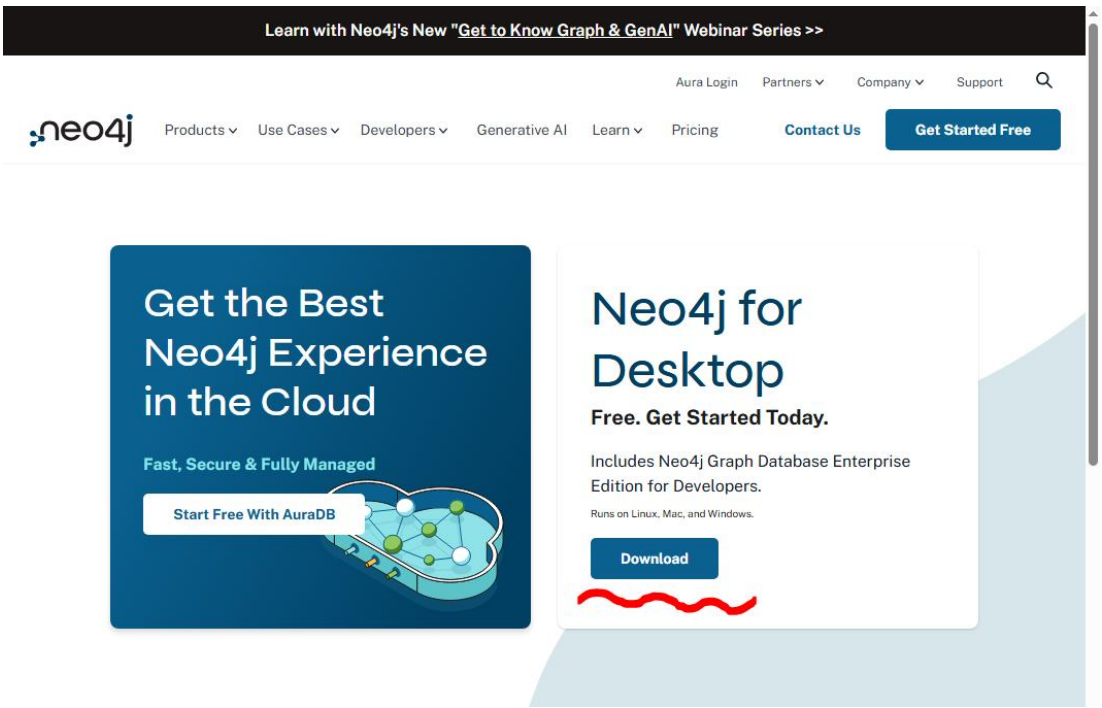
Реалізувати запити на пошук, додавання та видалення даних з навчальної бази даних **movies**, що поставляється разом з графовою СУБД Neo4j. Для цього виконати **Частину 1**, яка є **обов'язковою**. Реалізувати клієнтський застосунок (**Частина 2**), який знадобиться у наступній лабораторній роботі.

Примітки! Мова програмування (для обох рівнів), на якій буде розроблено клієнтський застосунок, обирається студентом самостійно.

ПОРЯДОК ВИКОНАННЯ

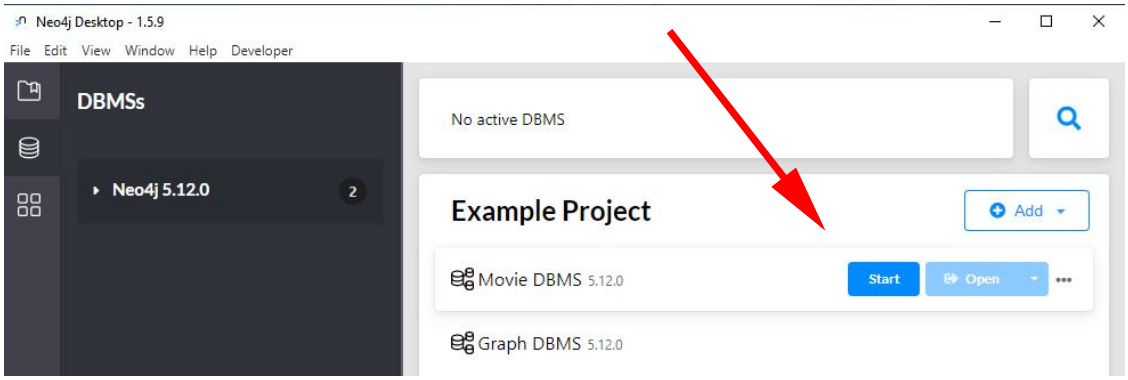
ЧАСТИНА 1 (оцінюється у **4 бали**).

1. Встановити СУБД Neo4j з офіційного сайту https://neo4j.com/download/ (для завантаження дистрибутиву вимагається реєстрація на сайті).

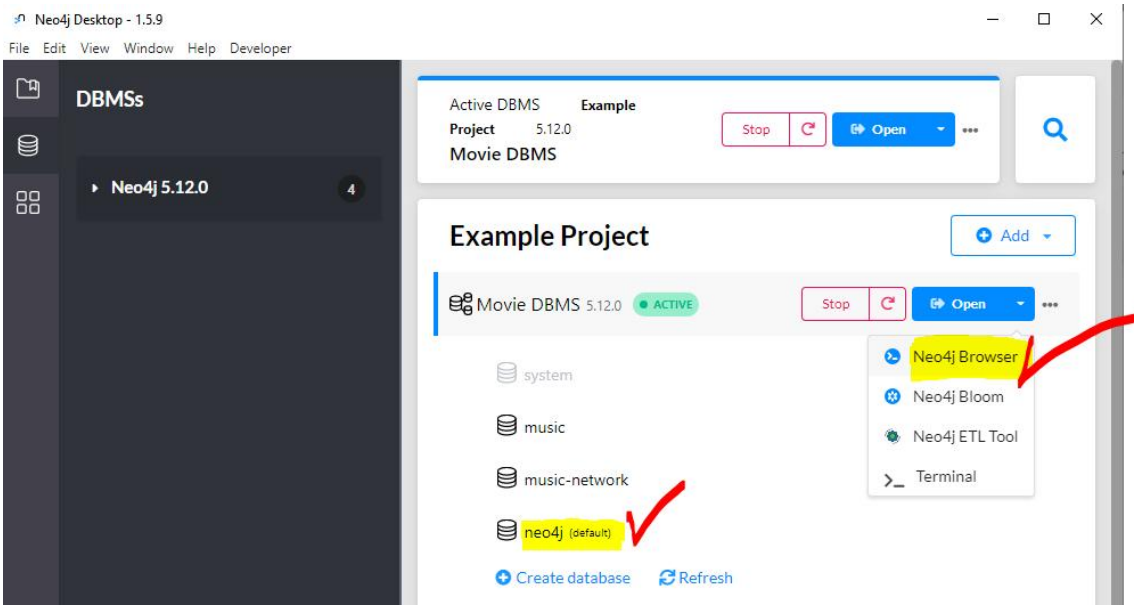


2. Використовуючи графічний інтерфейс Neo4j Browser (Neo4j Bloom) познайомитись зі структурою бази даних movies.

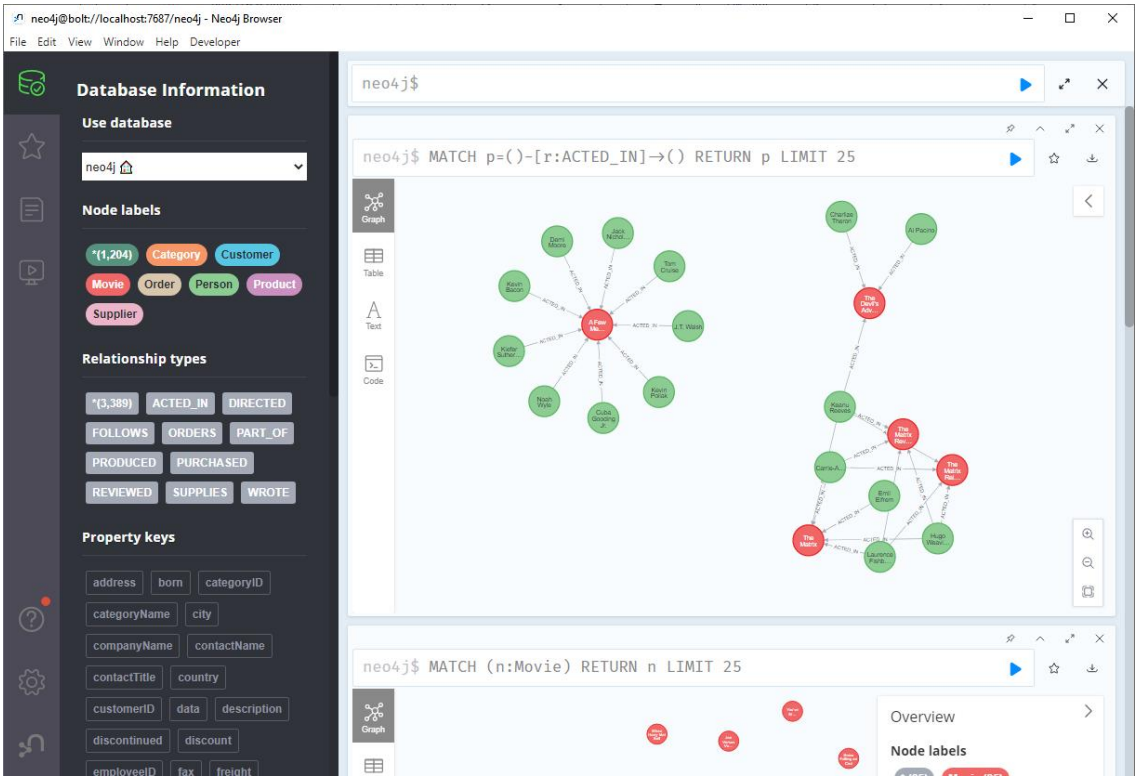
1)


Neo4j Desktop - 1.5.9
File Edit View Window Help Developer
DBMSs
Neo4j 5.12.0
2
No active DBMS
Example Project
Add
Movie DBMS 5.12.0
Start
Open
Graph DBMS 5.12.0

2)


Neo4j Desktop - 1.5.9
File Edit View Window Help Developer
DBMSs
Neo4j 5.12.0
4
Active DBMS Example
Project 5.12.0
Movie DBMS
Stop
Open
Example Project
Add
Movie DBMS 5.12.0
ACTIVE
Stop
Open
Neo4j Browser
Neo4j Bloom
Neo4j ETL Tool
Terminal
system
music
music-network
neo4j (default)
Create database
Refresh

3)


neo4j@bolt://localhost:7687/neo4j - Neo4j Browser
File Edit View Window Help Developer
Database Information
Use database
neo4j
Node labels
*(1,204)
Category
Customer
Movie
Order
Person
Product
Supplier
Relationship types
*(3,389)
ACTED_IN
DIRECTED
FOLLOWS
ORDERS
PART_OF
PRODUCED
PURCHASED
REVIEWED
SUPPLIES
WROTE
Property keys
address
born
categoryID
categoryName
city
companyName
contactName
contactTitle
country
customerID
data
description
discontinued
discount
neo4j$
neo4j$ MATCH p=()-[r:ACTED_IN]→() RETURN p LIMIT 25
Graph
Table
Text
Code
neo4j$ MATCH (n:Movie) RETURN n LIMIT 25
Overview
Node labels

Обираючи ліворуч ту чи іншу категорію Node labels та Relationship types, можна зрозуміти взаємозв'язки між різними сутностями, які присутні в цій базі даних.

3. Реалізувати наступні запити на пошук даних:

a) знайти всі фільми, реліз яких відбувся у 2003 році;
b) отримати всіх акторів, які знімались у фільмі The Matrix;
c) отримати всі фільми, в яких знімався Keanu Reeves;
d) отримати всіх акторів, з якими перетинався на знімальному майданчику Keanu Reeves (тобто зніалися в тих самих фільмах);
e) знайти людей, які одночасно виступали і в якості акторів і в якості режисерів (relationship type: DIRECTED) в одному і тому ж фільмі; якщо таких немає, то в різних фільмах;
f) Знайти актора (акторів), який знімався в найбільшій кількості фільмів;
g) Знайти найкоротший ланцюжок знайомства між акторами Keanu Reeves та Tom Cruise.

4. Реалізувати наступні запити на додавання даних:

a) додати до бази даних інформацію про свій улюбдений фільм (або один з найулюбленіших);
b) додати всіх акторів (якщо їх ще немає в базі) з цього фільму;
c) додати в базу даних режисера (relationship type: DIRECTED), продюсера (relationship type: PRODUCED), сценариста (relationship type: WROTE) цього фільму.

5. Реалізувати наступні запити на видалення даних:

a) видалити інформацію про якогось з акторів, доданих на попередньому кроці;
b) видалити всіх акторів, що знімалися у фільму, доданому в попередньому пункті,
c) видалити інформацію про фільм, доданий на попередньому кроці.

6. У звіті навести розроблені запити, результати їх виконання та відповіді на контрольні запитання.

ЧАСТИНА 2 (оцінюється **різною кількістю балів** (4 або 6) в залежності від обраного рівня).

Базовий рівень (оцінюється у **4 бали**)

1. Розробити клієнтський застосунок з графічним інтерфейсом користувача, який буде дозволяти користувачеві:

a) здійснювати вибір фільму, щоб подивитись перелік акторів, режисера, сценариста та продюсера цього фільму, рік виходу фільму на екрани тощо;
b) зійснювати вибір актора, щоб подивитись перелік фільмів, в яких він грав;

c) Здійснювати вибір режисера (продюсера), щоб подивитись, з якими фільмами він працював;

d) за обраним фільмом здійснювати вибір всіх інших фільмів, які були зняті тим самим режисером/режисерами (продюсером/продусерами);

e) (опціонально) додавати до інформаційної системи інформацію про фільми/акторів/продюсерів/режисерів/сценаристів;

f) (опціонально) реалізовувати якісь інші запити до бази даних movie (пошук найбільш затребуваного актора/продюсера/режисера тощо).

2. Доповнити звіт кодом застосунку та скріншотами, що демонструють його роботу.

Підвищений рівень (оцінюється у **6 балів**)

1. Розробити клієнт-серверну систему, що імітує веб-додаток, використовуючи сокети для зв'язку між клієнтською та серверною частинами, Redis для обмеження частоти запитів (Rate Limiting) та Neo4j як основну базу даних для зберігання та обробки даних (рис. 1).

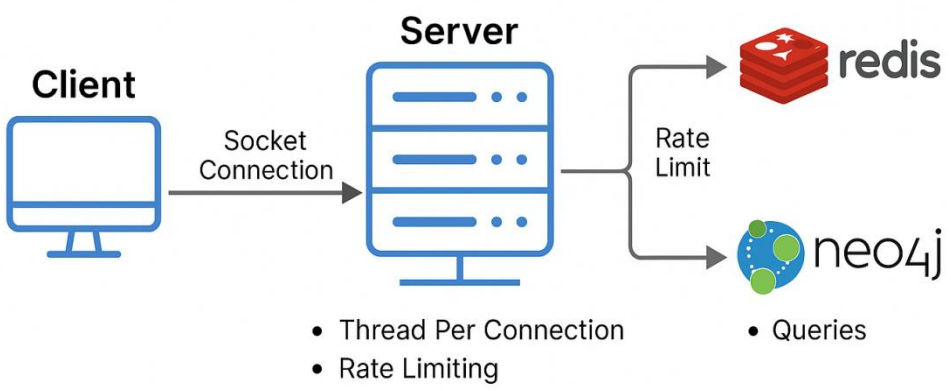


Рис. 1. Архітектура застосунку

a) Розробити серверний застосунок, в якому створити слухаючий сокет на визначеному порту.

b) Для кожного нового з'єднання створювати новий потік та передавати йому сокет клієнта для подальшої обробки.

c) Реалізувати контроль кількості запитів в одиницю часу від одного клієнта (Rate Limiting з "ковзним вікном") з використанням Redis:

- ідентифікацію клієнта можна проводити, наприклад, за IP-адресою, отриманою із сокета, або за певним штучним id, який передається у якості параметру (щоб мати можливість “розрізняти” клієнтів, які, у випадку використання одного компьютера в межах лабораторної роботи, будуть мати однакову IP-адресу); я
- якщо ліміт не перевищено, оновлювати лічильник у Redis та продовжити обробку запиту;

– якщо ліміт перевищено, відправити клієнту повідомлення про помилку ("Too Many Requests") та, можливо, закрити з'єднання.

Примітка! Приклад використання Redis для реалізації Rate Limiting наведено в матеріалах лекцій по Redis.

d) Розробити клієнтський застосунок, який буде звертатись до сервера із запитами на отримання даних з бази даних movie.

2. Доповнити звіт кодом застосунку та скріншотами, що демонструють його роботу.

Примітка! Приклад реалізації застосунку буде/було розглянуто на лекціях.

КОНТРОЛЬНІ ЗАПИТАННЯ

1. До якого типу NoSQL СУБД відноситься Neo4j? Які ще СУБД з цього класу є популярними на поточний момент? Аргументуйте свою відповідь, прикріпивши посилання на відповідні рейтинги.
2. В якому вигляді графові СУБД зберігають дані?
3. Яка мова запитів використовується в СУБД Neo4j?
4. Які типові варіанти застосування графових СУБД?
5. З чим пов'язана досить висока швидкість виконання операцій пошуку найкоротшого шляху графових СУБД у порівнянні з реляційними?
6. Які основні недоліки можна виділити для графових СУБД?
7. Чи підтримує Neo4j механізм транзакцій?
8. Чи підтримує Neo4j механізм реплікацій та шардингу?

Приклад підключення до Neo4j на Python:

```
import neo4j

DATABASE_NAME = "neo4j"
URI = "bolt://localhost:7687"
USERNAME = "neo4j"
PASSWORD = "password"

driver = None
try:
    driver = neo4j.GraphDatabase.driver(URI, auth=(USERNAME, PASSWORD))
    with driver.session(database=DATABASE_NAME) as session:
        query = "MATCH (m:Movie) RETURN m"
        result = session.run(query)
        for record in result:
            print(f"Movie title: {record["m"]["title"] }")

except Exception as e:
```

```
    print(f"Connection or request error: {e}")
finally:
    if driver:
        driver.close()
```